

Implications of quantum automata for contextuality

Jibran Rashid^{1,*} and Abuzer Yakaryılmaz^{2,**}

¹ Facoltà di Informatica, Università della Svizzera Italiana, Via G. Buffi 13, 6900, Lugano, Switzerland

² University of Latvia, Faculty of Computing, Raina bulv. 19, Rīga, LV-1586, Latvia
jibran.rashid@usi.ch, abuzer@lu.lv

Abstract. We construct zero-error quantum finite automata (QFAs) for promise problems which cannot be solved by bounded-error probabilistic finite automata (PFAs). Here is a summary of our results:

1. There is a promise problem solvable by an exact two-way QFA in exponential expected time, but not by any bounded-error sublogarithmic space probabilistic Turing machine (PTM).
2. There is a promise problem solvable by an exact two-way QFA in quadratic expected time, but not by any bounded-error $o(\log \log n)$ -space PTMs in polynomial expected time. The same problem can be solvable by a one-way Las Vegas (or exact two-way) QFA with quantum head in linear (expected) time.
3. There is a promise problem solvable by a Las Vegas realtime QFA, but not by any bounded-error realtime PFA. The same problem can be solvable by an exact two-way QFA in linear expected time but not by any exact two-way PFA.
4. There is a family of promise problems such that each promise problem can be solvable by a two-state exact realtime QFAs, but, there is no such bound on the number of states of realtime bounded-error PFAs solving the members this family.

Our results imply that there exist zero-error quantum computational devices with a *single qubit* of memory that cannot be simulated by any finite memory classical computational model. This provides a computational perspective on results regarding ontological theories of quantum mechanics [19], [30]. As a consequence we find that classical automata based simulation models [23], [6] are not sufficiently powerful to simulate quantum contextuality. We conclude by highlighting the interplay between results from automata models and their application to developing a general framework for quantum contextuality.

1 Preliminaries

Consider Alice and Bob who are presented with a 3×3 grid as depicted in Figure 1. They are asked to determine entries $A_i \in \{-1, +1\}$ for each cell in the

* Jibran Rashid was supported by QSIT Director's Reserve Project.

** Abuzer Yakaryılmaz was partially supported by ERC Advanced Grant MQC.

grid such that the parity for each row and column is “+1” except for the third column which has parity “−1”. Let R_i be the parity for row i and C_j be the parity for column j . The fact that no such assignment exists for the square can be verified by noting that $\prod_{i=1}^3 R_i = 1$ while $\prod_{j=1}^3 C_j = -1$.

	C1	C2	C3		
R1	A ₁	A ₂	A ₃	1	Z⊗I
R2	A ₄	A ₅	A ₆	1	I⊗X
R3	A ₇	A ₈	A ₉	1	Z⊗X
	1	1	−1	?	X⊗Z
					Y⊗Y

Fig. 1. Peres–Mermin magic square on the left. Each entry in the right square gives the measurement performed by the players to generate the corresponding output bit for the Peres–Mermin square.

After determining a common strategy, the players are spatially separated and the game proceeds as follows. Alice receives input i and Bob receives input j , each chosen uniformly random from the set $\{1, 2, 3\}$. They are required to output cell entries corresponding to row i and column j respectively, such that the parity requirement is satisfied and furthermore the common cell in their output is consistent, i.e., both of them assign it the same value. The game is called the Peres–Mermin magic square [33], [27] and is an example of the more general Kochen–Specker theorem [24].

Even though no classical strategy allows the players to win the magic square game with certainty, if the players share a pair of Bell states given by

$$|\psi\rangle = \frac{1}{\sqrt{2}} (|00\rangle + |11\rangle)^{\otimes 2},$$

then performing the measurements given in Figure 1 result in correlations that always satisfy the magic square requirements. This does not correspond to a fixed assignment to the square, just that in each independent run of the game, Alice and Bob are able to generate output that satisfies the requirements imposed on the rows and columns. This behaviour is usually studied under the notion of *contextuality* in quantum mechanics.

Recent work has focused on developing a general framework for contextuality based on generating a hypergraph for a given contextuality scenario and studying its combinatorial properties [12],[1]. Even though graph theoretic structures are appropriate for modelling contextuality they lack the computational perspective that comes from modelling the computational procedures that

generate contextuality scenarios. Quantum automata provide exactly such a framework. As a direct consequence of such considerations, we find that separations between classical and quantum finite automata imply that no amount of finite memory is in general sufficient to simulate quantum behaviour. Similar results have also been obtained by Hardy [19] and Montina [30].

Kleinmann et al. [23] and Blasiak [6] have suggested a classical simulation of Peres-Mermin magic square using classical memory. Cabello and Joosten [11] have shown that the amount of memory required to simulate the measurement results of the generalized Peres-Mermin square increasingly violate the Holevo bound. Cabello [9] proposed the *principle of bounded memory* which states that the memory a finite physical system can keep is bounded. On the other hand, Cabello [8] has also shown that the memory required to produce quantum predictions grows at least exponentially with the number of qubits n .

We show that a stronger statement follows from our results on the separations between quantum and classical finite automata. More specifically, there exist promise problems that quantum automata equipped with a *single qubit* can solve with zero-error while no classical finite memory model can solve these problems with bounded error. In contrast, the exponential separation obtained by Cabello [8] requires a quantum system of size n . The hidden variable model for a single qubit due to Bell [5] does not apply since there are only finite bits available for the classical simulation.

We assume the reader is familiar with the basics of quantum computation [32] and the basic models in automata theory [38].

1.1 Quantum Automata

We denote input alphabet by Σ , which does not include $\$$ (the left end-marker), $\$$ (the right end-marker) and $\bar{\Sigma} = \Sigma \cup \{\$, \$\}$. A promise problem is a pair $P = (P_{\text{yes}}, P_{\text{no}})$, where $P_{\text{yes}}, P_{\text{no}} \subseteq \Sigma^*$ and $P_{\text{yes}} \cap P_{\text{no}} = \emptyset$ [37]. P is said to be solved by a machine \mathcal{M} with error bound $\epsilon \in (0, \frac{1}{2})$ if any member of P_{yes} is accepted with a probability at least $1 - \epsilon$ and any member of P_{no} is rejected by \mathcal{M} with a probability at least $1 - \epsilon$. P is said to be solved by \mathcal{M} with bounded-error if it is solved by \mathcal{M} with an error bound. If $\epsilon = 0$, then it is said that the problem is solved by \mathcal{M} *exactly*. A special case of bounded-error is one-sided bounded-error where either all members of P_{yes} are accepted with probability 1 or all members of P_{no} are rejected with probability 1. \mathcal{M} is said to be Las Vegas (with a success probability $p \in (0, 1]$) [21] if

- \mathcal{M} has the ability of giving three answers (instead of two): “accept”, “reject”, or “don’t know”;
- for a member of P_{yes} , \mathcal{M} gives the decision of “acceptance” with a probability at least p and gives the decision of “don’t know” with the remaining probability; and,
- for a member of P_{no} , \mathcal{M} gives the decision of “rejection” with a probability at least p and gives the decision of “don’t know” with the remaining probability.

If P satisfies $P_{\text{yes}} \cup P_{\text{no}} = \Sigma^*$ and it is *solvable* by \mathcal{M} , then it is conventional said that P_{yes} is *recognized* by \mathcal{M} .

All models in this paper have a single-head read-only tape on which the given input string is placed between left and right end-markers. The head never leaves the end-markers. The input head can move to the left, move to the right, or stay on the same square. This property is denoted as “two-way”. If the input head is not allowed to move to left, then it is called “one-way”. As a further restriction, if the input head is allowed to stay on the same square only for a fixed-number of steps, then it is called “realtime”. Note that any realtime quantum model has a classical head.

A two-way automaton is called sweeping if the input head is allowed to change its direction only on the end-markers [34,22]. A very restricted version of sweeping automaton called restarting realtime automaton runs a realtime algorithm in an infinite loop, [41], i.e. if the computation is not terminated on the right end-marker, the same realtime algorithm is executed again.

A two-way finite automaton with quantum and classical states (2QCFA) [3] is a two-way deterministic finite automaton augmented with a fixed-size quantum register. Formally,³ a 2QCFA is

$$\mathcal{M} = (S, Q, \Sigma, \delta, s_1, q_1, s_a, s_r),$$

where S and Q are the set of classical and quantum states, respectively; $s_1 \in S$ and $q_1 \in Q$ are initial classical and quantum states, respectively; $s_a \in S$ and $s_r \in S$ ($s_a \neq s_r$) are the accepting and rejecting states, respectively; and δ is the transition function composed by two sub-elements δ_q and δ_c that govern the quantum part and classical part of the machine, respectively. Suppose that \mathcal{M} is in state $s \in S$ and the symbol under the input head is $\sigma \in \Sigma$. In each step, first the quantum part and then the classical part is processed in the following manner:

- $\delta_q(s, \sigma)$ determines either a unitary operator, say $U_{s, \sigma}$, or a projective operator, say $P_{s, \sigma} = \{P_{s, \sigma, 1}, \dots, P_{s, \sigma, k}\}$ for some $k > 0$, and then it is applied to the quantum register. Formally, in the former case,

$$\delta_q(s, \sigma, |\psi\rangle) \rightarrow (i = 1, U|\psi\rangle),$$

and, in the latter case,

$$\delta_q(s, \sigma, |\psi\rangle) \rightarrow \left\{ \left(i, \frac{|\psi_i\rangle}{\sqrt{\langle \psi_i | \psi_i \rangle}} \right) \middle| |\psi_i\rangle = P_{s, \sigma, i} |\psi\rangle, \langle \psi_i | \psi_i \rangle \neq 0, \text{ and } 1 \leq i \leq k. \right\}.$$

We fix $i = “1”$ if a unitary operator is applied. Note that only a single outcome ($i \in \{1, \dots, k\}$) can be observed in the case of a projective measurement.

³ Here, we define a slightly different model than the original one, but, they can simulate each other exactly.

- After the quantum phase, the machine evolves classically. Formally,

$$\delta(s, \sigma, i) \rightarrow (s', d),$$

where i is the measurement outcome of quantum phase, s' is the new classical state, and $d \in \{\leftarrow, \downarrow, \rightarrow\}$ represents the update of the input head.

Note that, for Las Vegas algorithms, we need to define another halting state called s_d corresponding to answer “don’t know”.

The computation of \mathcal{M} on a given input string w starts in the initial configuration, where the head is on the first symbol of $\tilde{w} = \$w\$$, the classical state is s_1 , and the quantum state is $|q_1\rangle$. The computation is terminated and the input is accepted (resp., rejected) if \mathcal{M} enters to state s_a (resp., s_r).

A 2QCFA restricted to a realtime head is denoted by rtQCFA. Formally, defined in [43], on each tape square a rtQCFA applies a unitary operator followed by a projective measurement, and then evolves its classical part.⁴ Another restricted realtime QFA model is the Moore-Crutchfield quantum finite automaton (MCQFA) [31]. It consists of only quantum states and a single unitary operator determined by the scanned symbol is applied on each tape square. A projective measurement is applied at the end of computation. A probabilistic or quantum automaton is called rational or algebraic if all the transitions are restricted to rational or algebraic numbers.

2 Quantum automata for promise problems

In this section, we present some promise problems solvable by QFAs *without error* but not solvable by their *bounded-error* probabilistic counterparts. At the end, we will also show that the family of promise problem, which was shown to be solvable by a family of exact realtime QFAs (MCQFAs) having only two states [4], cannot be solvable by a family of bounded-error probabilistic finite automata (PFAs) having a fixed number of states.

2.1 Exact rational (sweeping) 2QCFA algorithm

2QCFA can recognize $\text{PAL} = \{w \mid w \in \{a, b\}^* \text{ and } w = w^r\}$ for any one-sided error bound [3,41]. In the case of one-sided error, one decision is always reliable. We use this fact to develop quantum automata for solving promise problems inherited from PAL . We know that PAL cannot be recognized by bounded-error PTMs using sublogarithmic space [15,18]. We take into consideration these facts when formulating our promise problems so that the impossibility results for bounded-error probabilistic algorithms are still applicable for our constructions.

Our first promise problem is $\text{PromisePAL} = (\text{PromisePAL}_{\text{yes}}, \text{PromisePAL}_{\text{no}})$, where

⁴ This definition is sufficient to obtain the most general realtime quantum finite automaton [20,42]. Moreover, allowing more than one quantum or classical transition on the same tape square does not increase the computational power of rtQCFA.

- $\text{PromisePAL}_{\text{yes}} = \{ucv | u, v \in \{a, b\}^*, |u| = |v|, u \in \text{PAL}, \text{ and } v \notin \text{PAL}\}$ and
- $\text{PromisePAL}_{\text{no}} = \{ucv | u, v \in \{a, b\}^*, |u| = |v|, u \notin \text{PAL}, \text{ and } v \in \text{PAL}\}$.

Each of the two 2QCFA algorithms given for PAL in [3] and [41] have zero-error when they reject. That is, for a given $\epsilon \in (0, \frac{1}{2})$, there exists a 2QCFA \mathcal{M}_ϵ which always accepts every string $w \in \text{PAL}$ and every $w \notin \text{PAL}$ is accepted with probability at most ϵ and it is rejected with probability at least $1 - \epsilon$. So, if \mathcal{M}_ϵ rejects an input, we can be certain that the input is a non-member.

We can design an exact 2QCFA, say $\mathcal{EXACT}_{\mathcal{PAL}}$, for **PromisePAL** based on \mathcal{M}_ϵ as follows: Let $w = ucv \in \text{PromisePAL}$ be the input such that $u, v \in \{a, b\}^*$ and $|u| = |v|$. On input string w , $\mathcal{EXACT}_{\mathcal{PAL}}$ proceeds in an infinite loop as follows,

- the computation splits into two branches on the left end-marker with probabilities $\frac{16}{25}$ and $\frac{9}{25}$, respectively, by applying a rational unitary operator to a qubit followed by a measurement in the computational basis;
- in the 1st branch, $\mathcal{EXACT}_{\mathcal{PAL}}$ executes \mathcal{M}_ϵ on v and *accepts* w if $\mathcal{EXACT}_{\mathcal{PAL}}$ *rejects* v ;
- in the 2nd branch, $\mathcal{EXACT}_{\mathcal{PAL}}$ executes \mathcal{M}_ϵ on u and *rejects* w if $\mathcal{EXACT}_{\mathcal{PAL}}$ *rejects* u ; and,
- the computation continues, otherwise.

Note that only a single decision is given in each branch: In the 1st branch, the members of $\text{PromisePAL}_{\text{yes}}$ are accepted with a probability at least $1 - \epsilon$ and no decision is given on the members of $\text{PromisePAL}_{\text{no}}$. In the 2nd branch, no decision is given on the members of $\text{PromisePAL}_{\text{yes}}$ and the members of $\text{PromisePAL}_{\text{no}}$ are rejected with a probability at least $1 - \epsilon$. Thus, in a single round, the members of $\text{PromisePAL}_{\text{yes}}$ are accepted with a probability at least $\frac{16}{25}(1 - \epsilon)$ and the members of $\text{PromisePAL}_{\text{no}}$ are rejected with a probability at least $\frac{9}{25}(1 - \epsilon)$. Thus, $\mathcal{EXACT}_{\mathcal{PAL}}$ separates $\text{PromisePAL}_{\text{yes}}$ and $\text{PromisePAL}_{\text{no}}$ exactly by calling \mathcal{M}_ϵ in expected linear time. This establishes Theorem 1 while the fact that sublogarithmic space PTMs cannot solve **PromisePAL** is established in Theorem 2, proof of which can be found in Appendix A.

Theorem 1. *PromisePAL can be solvable by an exact rational sweeping 2QCFA in exponential expected time.*

Theorem 2. *Bounded-error sublogarithmic space PTMs cannot solve PromisePAL.*

The scheme given above can be easily generalized to many other cases. The size of the quantum register, the type of the head, and the type of the transitions are determined by \mathcal{M}_ϵ . Specifically, (i) if \mathcal{M}_ϵ is restarting (sweeping), then $\mathcal{EXACT}_{\mathcal{PAL}}$ is restarting (sweeping), too, or (ii) if \mathcal{M}_ϵ has only rational (algebraic) amplitudes, then $\mathcal{EXACT}_{\mathcal{PAL}}$ has rational (algebraic) amplitudes.

The 2QCFA algorithm for PAL given by Ambainis and Watrous [3] is rational and sweeping. The one given by Yakaryılmaz and Say in [41] is restarting but uses algebraic numbers. Both of them run in expected exponential time. In

the next subsection we present a new promise problem and we use the former algorithm to obtain an exact rational restarting rtQCFA. Currently, we do not know how to obtain a similar result based on the latter model except by utilizing superoperators.

2.2 Exact rational restarting rtQCFA algorithm

Now, we define a promise problem (a modified version of **PromisePAL**) solvable by an exact rational restarting rtQCFA but not by any sublogarithmic space PTMs: $\text{PromiseTWINPAL} = (\text{PromiseTWINPAL}_{\text{yes}}, \text{PromiseTWINPAL}_{\text{no}})$, where

- $\text{PromiseTWINPAL}_{\text{yes}} = \{ucucvcv|u, v \in \{a, b\}^+, |u| = |v|, u \in \text{PAL}, \text{ and } v \notin \text{PAL}\}$, and
- $\text{PromiseTWINPAL}_{\text{no}} = \{ucucvcv|u, v \in \{a, b\}^+, |u| = |v|, u \notin \text{PAL}, \text{ and } v \in \text{PAL}\}$.

Theorem 3. *There is an exact rational restart rtQCFA that solves **PromiseTWINPAL** in exponential expected time. (See Appendix B for the proof)*

Theorem 4. ***PromiseTWINPAL** cannot be solved by any bounded-error $o(\log n)$ -space PTM. (See Appendix C for the proof)*

2.3 Las Vegas rational rtQCFA algorithm

Here, we present another promise problem solvable by Las Vegas rtQCFAs or linear-time exact 2QCFAs but not by any bounded-error realtime PFA (rtPFA). Since any exact 2PFA can be simulated by a realtime deterministic finite automaton (rtDFA) (Appendix D), exact two-way PFAs (2PFAs) also cannot solve this new promise problem. The new promise problem is given by: $\text{EXPPromiseTWINPAL} = (\text{EXPPromiseTWINPAL}_{\text{yes}}, \text{EXPPromiseTWINPAL}_{\text{no}})$, where

- $\text{EXPPromiseTWINPAL}_{\text{yes}} = \{(ucucvcvc)^t|u, v \in \{a, b\}^+, |u| = |v|, u \in \text{PAL}, v \notin \text{PAL}, \text{ and } t \geq 25^{|u|}\}$, and
- $\text{EXPPromiseTWINPAL}_{\text{no}} = \{(ucucvcvc)^t|u, v \in \{a, b\}^+, |u| = |v|, u \notin \text{PAL}, v \in \text{PAL}, \text{ and } t \geq 25^{|u|}\}$.

Theorem 5. ***EXPPromiseTWINPAL** can be solved by a Las Vegas rational rtQCFA or by an exact rational restarting rtQCFA in linear expected time. (See Appendix E for the proof.)*

Theorem 6. *There is no bounded-error rtPFA that solves **EXPPromiseTWINPAL**. (See Appendix F for the proof.)*

2.4 Polynomial-expected-time algebraic exact restarting rtQCFA algorithm

Our promise problem for this section is as follows:

$$\text{PromiseEQ} = (\text{PromiseEQ}_{\text{yes}}, \text{PromiseEQ}_{\text{no}}), \text{ where}$$

$$\text{PromiseEQ}_{\text{yes}} = \{a^m b a^m b a^n | m \neq n\} \text{ and } \text{PromiseEQ}_{\text{no}} = \{a^m b a^n b a^m | m \neq n\}.$$

Theorem 7. *There is an exact algebraic restarting rtQCFA algorithm solving PromiseEQ in quadratic expected time. Moreover, PromiseEQ can be solved by a Las Vegas rational one-way QFA in linear time or by an exact rational two-way QFAs in linear expected time, where both models have a quantum head [25,39]. (See Appendix G for the proof)*

Theorem 8. *PromiseEQ cannot be solved by any bounded-error $o(\log \log n)$ -space PTMs in sub-exponential expected time. (See Appendix H for the proof)*

2.5 Succinctness of realtime QFAs

For a given positive integer k , $\text{EVENODD}^k = (\text{EVENODD}_{\text{yes}}^k, \text{EVENODD}_{\text{no}}^k)$ is a promise problem [4] such that

- $\text{EVENODD}_{\text{yes}}^k = \{a^{i2^k} \mid i \text{ is a nonnegative even integer}\}$, and
- $\text{EVENODD}_{\text{no}}^k = \{a^{i2^k} \mid i \text{ is a nonnegative odd integer}\}$.

Ambainis and Yakaryılmaz [4] showed that EVENODD^k can be solved by a 2-state MCQFA exactly, but, the corresponding probabilistic automaton needs at least 2^{k+1} states. We show in Theorem 9 that allowing errors in the output does not help in decreasing the space requirement. Proof of Theorem 9 is provided in Appendix I.

Theorem 9. *Bounded-error rtPFAs need at least 2^{k+1} states to solve EVENODD^k .*

3 Noncontextual inequalities from automata

We begin by reformulating the Peres-Mermin game in terms of inequalities. Let $\langle A_i A_j A_k \rangle$ be the expected parity of the corresponding entries of the square. We associate with each strategy, a value of the game $\langle \chi \rangle$, which is given by

$$\begin{aligned} \langle \chi \rangle = & \langle A_1 A_2 A_3 \rangle + \langle A_4 A_5 A_6 \rangle + \langle A_7 A_8 A_9 \rangle \\ & + \langle A_1 A_4 A_7 \rangle + \langle A_2 A_5 A_8 \rangle - \langle A_3 A_6 A_9 \rangle. \end{aligned}$$

The classical bound is $\langle \chi \rangle \leq 4$, while the quantum bound is given by $\langle \chi \rangle \leq 6$. We can now construct similar inequalities for the promise problems defined in this paper. The general idea behind the inequalities is to construct a game based on quantum and classical automata separations. Assume Bob is restricted to either N bits of classical memory or N quantum bits and Alice has the task of

verifying what type of memory is available to Bob. She can query Bob multiple times on a pre-selected problem that is known to both of them. Conditioned on the classical memory requirement for the problem the idea then is for Alice to iteratively query Bob on input strings of increasing length. Eventually Bob's classical memory becomes insufficient to correctly answer the query and his best response is a random guess.

PAL and PromisePAL can be solved in log space. As a consequence, Alice requires an exponential number of queries in N before Bob's memory is exhausted. The classical exponential memory requirement for EVENODD^k means that number of queries need only be logarithmic before a violation is observed. On the other hand, EVENODD^k is not a single problem but a family of promise problems and the classical memory requirement is for rtPFAs. For PromisePAL we obtain zero error for the quantum strategy while for the classical strategy bounded-error is not possible for 2PFAs.

We base the inequality we present on EVENODD^k . The arguments carry over to PAL and PromisePAL as well. On a given query Bob receives as input an integer k and a unary string $w = a^l$ that is promised to be from either $\text{EVENODD}_{\text{yes}}^k$ or $\text{EVENODD}_{\text{no}}^k$, i.e., $l = i2^k$. The task for Bob is to determine the membership of string w , i.e., whether i is even or odd. So, he outputs “+1” if i is even and “−1” otherwise. The identification can be made for any k if Bob has unbounded memory. If Bob is restricted to have memory 2^{n+1} then the identification can still be made perfectly for all integer inputs to Bob with $k \leq n$. It becomes impossible to perform this identification perfectly when $k > n$. In this case the amount of memory available to Bob is not sufficient to determine classically the membership of the input string w .

In the quantum case, a perfect strategy exists for all k , if Bob is allowed access to a single qubit $|\psi\rangle$. The state is initialized to $|0\rangle$ and for each “ a ” in the string w Bob applies the rotation $U_a = \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix}$, with $\theta = \frac{\pi}{2^{k+1}}$. Bob measures in the computational basis once the input is processed and realizes that i is odd if he obtains $|1\rangle$ or $-|1\rangle$ and i is even is the result is $|0\rangle$ or $-|0\rangle$. This procedure guarantees that Bob will always correctly identify the string w .

Assume that the amount of memory available to Bob is N but we do not know N . Let $\langle A_y^k \rangle$ be the expected value of Bob's output when the input is k and i is even. Similarly, $\langle A_n^k \rangle$ represents the expected value for input k and i odd. One way to verify whether Bob is quantum or classically memory bound is to initially query for a choice of k and then sequentially query for increasing size of k . For each choice of k we choose i to be odd or even with a uniform random distribution. We define V to measure how successful Bob is in correctly identifying the string w . Performing the procedure Q times gives us a value

$$V = \sum_{j=1}^Q \langle A_y^{k=4j} \rangle - \langle A_n^{k=4j} \rangle, \quad (1)$$

where we have chosen to increase the input k by multiples of 4 at each iteration. If $k \leq \log N - 1$, then for both the classical and quantum case Bob can achieve

$V = Q$. For $k > \log N - 1$, since there is no perfect strategy in the classical case we have $V < Q$, while the quantum strategy still achieve $V = Q$. The classical value can be made much tighter since the optimal classical strategy for $k > \log N - 1$ is just a random guess. We have shown in Section 2.5 that allowing error classically does not help in terms of reducing the memory requirement, i.e. bounded-error rtPFAs need at least 2^{k+1} states to solve **EVENODD**^k. This implies that the classical value is bounded by $\frac{\log N - 1}{4}$. Similar inequalities may be derived for **PAL** and **PromisePAL** and they are summarized in Tables 1 and 2.

Problem	Type	Classical Memory	Inequality Value
PAL	language recognition	$\log n$	$\frac{2^N}{4}$
EVENODD ^k	family of promise problems	2^{k+1}	$\frac{\log N - 1}{4}$
PromisePAL	promise problem	$\log n$	$\frac{2^N}{4}$

Table 1. For both **PAL** and **PromisePAL** no 2PFA exists that solves the problem with bounded error. For the family of $\{\text{EVENODD}^k \mid k > 0\}$, there is no bound on the number of states for real-time PFAs that solve the members of this family with bounded error. Given an input string of size n and classical memory N , the table gives the memory requirement for solving the specific instance and the value attained for the non-contextual inequality.

Problem	Quantum Model	Quantum Memory	Quantum Value
PAL	2QCFA	qubit	$Q - \delta$
EVENODD ^k	Real-time	qubit	Q
PromisePAL	2QCFA	qubit	Q

Table 2. The weakest known quantum models that solve the given problems and the associated error in the solution. The value attained for the inequality is related to the number of runs Q of the game.

It may be possible to improve these inequalities by finding other problems for which we obtain a similar separation as **PromisePAL** but with an exponential classical memory requirement and a polynomial time quantum automata.

4 Discussion

Perhaps the most alluring charm of quantum automata separations is the possibility they offer of constructing a computational device, that could solve a problem which no classic device with finite memory could. Rather than just ruling out hidden variables with an exponential size increase, these computational devices could be used in principle to rule out arbitrary size hidden variables by increasing the problem input size. The thorny issue though is a trade-off between the memory utilized and the amount of precision required in the interactions with the quantum memory. This precision requirement appears either in the form of the matrix entries for the unitaries, as in the case of **EVENODD**^k or in form of the ability to resolve two states that may be arbitrarily close to each other, as in **PAL**.

Any experimental setup is always restricted by some level of precision. If not by technological limitations then by more fundamental restrictions such as the uncertainty principle. The question that we are inevitably led to consider following this line of reasoning is whether it is possible to retain the quantum advantage in the automata model while still requiring finite precision in our interactions with the memory.

The intersection of ideas from classical simulation of contextuality and automata theory leads us to the notion of *Finite Precision Quantum Automata* (FPQA). A FPQA satisfies in addition to the usual automata requirements the constraints that for any two unique unitaries U_i and U_j applied during the computation we have $|U_i - U_j| > \epsilon$ and for any two different states $|\psi\rangle$ and $|\phi\rangle$ obtained during the computation we have $|\langle\psi|\phi\rangle|^2 > \delta$.

It is not clear whether we can construct a FPQA that still manages to provide a computational advantage over classical automata. Meyer [28] has argued that the Kochen-Specker theorem [24] does not hold when only finite precision measurements are available. Clifton and Kent [13] have generalized the arguments of Meyer for POVM's. On the other hand Mermin and Cabello [7] have independently argued that such nullification theorems do not hold. Recently Cabello and Cunha [10] have proposed a two-qutrit contextuality test, claiming it to be free of the finite precision loophole. These tests though do admit a finite memory simulation model. Constructing a FPQA that yields a separation over the classical models would not admit a finite memory simulation model and consequently provide a stronger separation.

Even if the FPQA model does turn out to be equivalent to the classical model in terms of the class of problems it solves, it does not take away from the succinctness advantage of the quantum model. In the previous section we argued that quantum automata separations serve as witnesses for distinguishing between genuinely quantum and space bounded classical players. We can flip the reasoning around and observe that simulating quantum contextuality is an inherently classical memory intensive task. This difficulty can be used to construct classical Proofs of Space as identified by Dziembowski et al. [17]. The idea is to establish that Bob has access to a certain amount of memory. Bob is asked to simulate an appropriately chosen quantum contextuality scenario. This would require exponential memory on Bob's side while the verifier could directly check that Bob's output satisfies the required quantum correlations. Note also that by definition, pre-computation does not allow Bob to simulate quantum contextuality. This task would require identification of generalizable quantum contextual promise problems. We note that use of the term 'contextual' here is different than the standard notion of context-free languages.

References

1. A. Acín, T. Fritz, A. Leverrier, and B. Sainz. A combinatorial approach to nonlocality and contextuality. 2014. <http://arxiv.org/abs/1212.4084>.

2. Andris Ambainis and Rūsiņš Freivalds. 1-way quantum finite automata: strengths, weaknesses and generalizations. In *FOCS'98: Proceedings of the 39th Annual Symposium on Foundations of Computer Science*, pages 332–341, 1998.
3. Andris Ambainis and John Watrous. Two-way finite automata with quantum and classical states. *Theoretical Computer Science*, 287(1):299–311, 2002.
4. Andris Ambainis and Abuzer Yakaryilmaz. Superiority of exact quantum automata for promise problems. *Information Processing Letters*, 112(7):289–291, 2012.
5. J. S. Bell. On the Einstein–Podolsky–Rosen paradox. *Physics*, 1:195, 1964.
6. P. Blasiak. Quantum cube: A toy model of a qubit. *Physics Letters A*, 377:847–850, 2013.
7. A. Cabello. Finite-precision measurement does not nullify the Kochen–Specker theorem. *Physical Review A*, 65:052101, 2002.
8. A. Cabello. *The Contextual Computer*, chapter 31, pages 595–604. 2012.
9. A. Cabello. The role of bounded memory in the foundations of quantum mechanics. *FP*, 42(1):68–79, 2012.
10. A. Cabello and M. T. Cunha. Proposal of a two-qutrit contextuality test free of the finite precision and compatibility loopholes. *Physical Review Letters*, 106:190401, 2011.
11. A. Cabello and J. J. Joosten. Hidden variables simulating quantum contextuality increasingly violate the Holevo bound. In *Unconventional Computation*, volume 6714 of *Lecture Notes in Computer Science*, pages 64–76. 2011.
12. A. Cabello, S. Severini, and A. Winter. Graph-theoretic approach to quantum correlations. *Physical Review Letters*, 112:040401, 2014.
13. R. Clifton and A. Kent. Simulating quantum mechanics by non-contextual hidden variables. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 456:2101–2114, 2000.
14. C. Dwork and L. Stockmeyer. A time complexity gap for two-way probabilistic finite-state automata. *SIAM Journal on Computing*, 19(6):1011–1123, 1990.
15. Cynthia Dwork and Larry Stockmeyer. Finite state verifiers I: The power of interaction. *Journal of the ACM*, 39(4):800–828, 1992.
16. Cynthia Dwork and Larry J. Stockmeyer. On the power of 2-way probabilistic finite state automata (extended abstract). In *FOCS'89: Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, pages 480–485, Research Triangle Park, North Carolina, 1989.
17. Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. Cryptology ePrint Archive, Report 2013/796, 2013. <http://eprint.iacr.org/>.
18. Rūsiņš Freivalds and Marek Karpinski. Lower space bounds for randomized computation. In *ICALP'94: Proceedings of the 21st International Colloquium on Automata, Languages and Programming*, pages 580–592, 1994.
19. L. Hardy. Quantum ontological excess baggage. *Studies in History and Philosophy of Science Part B: Studies in History and Philosophy of Modern Physics*, 35(2):267–276, 2004.
20. Mika Hirvensalo. Quantum automata with open time evolution. *International Journal of Natural Computing Research*, 1(1):70–85, 2010.
21. Juraj Hromkovic and Georg Schnitger. On the power of las vegas for one-way communication complexity, obdds, and finite automata. *Information and Computation*, 169(2):284–296, 2001.
22. Christos A. Kapoutsis, Richard Kráľovic, and Tobias Mömke. Size complexity of rotating and sweeping automata. *Journal of Computer and System Sciences*, 78(2):537–558, 2012.

23. M. Kleinmann, O. Gühne, J.R. Portillo, J.-Å. Larsson, and A. Cabello. Memory cost of quantum contextuality. 13(11):113011, 2011.
24. S. Kochen and E.P. Specker. The problem of hidden variables in quantum mechanics. *JMM*, 17:59–87, 1967.
25. Attila Kondacs and John Watrous. On the power of quantum finite state automata. In *FOCS'97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, pages 66–75, 1997.
26. Carlo Mereghetti, Beatrice Palano, and Giovanni Pighizzini. Note on the succinctness of deterministic, nondeterministic, probabilistic and quantum finite automata. *Theoretical Informatics and Applications*, 35(5):477–490, 2001.
27. N.D. Mermin. Simple unified form for the major no-hidden-variable theorems. *Physical Review Letters*, 65, 1990.
28. D.A. Meyer. Finite precision measurement nullifies the Kochen–Specker theorem. *PRL*, 83:3751–3754, 1999.
29. Massimiliano Milani and Giovanni Pighizzini. Tight bounds on the simulation of unary probabilistic automata by deterministic automata. *Journal of Automata, Languages and Combinatorics*, 6(4):481–492, 2001.
30. A. Montina. Exponential complexity and ontological theories of quantum mechanics. *Physical Review A*, 77:022104, 2008.
31. Cristopher Moore and James P. Crutchfield. Quantum automata and quantum grammars. *Theoretical Computer Science*, 237(1-2):275–306, 2000.
32. Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information*. Cambridge University Press, 2000.
33. A. Peres. Incompatible results of quantum measurements. 151:107–108, 1990.
34. Michael Sipser. Lower bounds on the size of sweeping automata. *Journal of Computer and System Sciences*, 21(2):195–202, 1980.
35. John Watrous. On the power of 2-way quantum finite state automata. Technical Report CS-TR-1997-1350, University of Wisconsin, 1997. (Also available at citeseer.ist.psu.edu/article/watrous97power.html).
36. John Watrous. *Space-bounded quantum computation*. PhD thesis, University of Wisconsin - Madison, USA, 1998.
37. John Watrous. Quantum computational complexity. In Robert A. Meyers, editor, *Encyclopedia of Complexity and Systems Science*, pages 7174–7201. Springer, 2009.
38. Abuzer Yakaryılmaz. *Classical and Quantum Computation with Small Space Bounds*. PhD thesis, Boğaziçi University, 2011. (arXiv:1102.0378).
39. Abuzer Yakaryılmaz. Superiority of one-way and realtime quantum machines. *RAIRO - Theoretical Informatics and Applications*., 46(4):615–641, 2012.
40. Abuzer Yakaryılmaz, Rūsiņš Freivalds, A. C. Cem Say, and Ruben Agadzanyan. Quantum computation with write-only memory. *Natural Computing*, 11(1):81–94, 2012.
41. Abuzer Yakaryılmaz and A. C. Cem Say. Succinctness of two-way probabilistic and quantum finite automata. *Discrete Mathematics and Theoretical Computer Science*, 12(2):19–40, 2010.
42. Abuzer Yakaryılmaz and A. C. Cem Say. Unbounded-error quantum computation with small space bounds. *Information and Computation*, 279(6):873–892, 2011.
43. Shenggen Zheng, Daowen Qiu, Lvzhou Li, and Jozef Gruska. One-way finite automata with quantum and classical states. In *Languages Alive*, volume 7300 of *LNCS*, pages 273–290, 2012.

A The proof of Theorem 2

We give a proof of the fact that 2PFAs cannot solve promise problem **PromisePAL** with bounded-error. We use the technique given in [15] that shows **PAL** cannot be recognized by bounded-error 2PFAs. We begin with the following fact.

Fact 1 (*Theorem 3.3 on Page 809 of [15]*)

Let $A, B \subseteq \Sigma^*$ with $A \cap B = \emptyset$. Suppose there is an infinite set I of positive integers and, for each $m \in I$, a set $W_m \subseteq \Sigma^*$ such that

1. $|x| \leq m$ for all $x \in W_m$,
2. for every integer k there is an m_k such that $|W_m| \geq m^k$ for all $m \in I$ with $m \geq m_k$, and
3. for every $m \in I$ and every $x, x' \in W_m$ with $x \neq x'$, there are strings $y, z \in \Sigma^*$ such that either $yxz \in A$ and $yx'z \in B$, or $yxz \in B$ and $yx'z \in A$.

Then no bounded-error 2PFA separates A and B .

Theorem 10. *Bounded-error 2PFAs cannot solve **PromisePAL**.*

Proof. Let $m > 0$ and $\{u_1, \dots, u_{2^m}\}$ be all strings in $\{a, b\}^m$. We define W_m as follows:

$$W_m = \{u_i c u_i \mid 1 \leq i \leq 2^m\}.$$

Since the size of W_m is super-polynomial in m , i.e., $|W_m| = 2^m$ we satisfy the second condition given in Fact 1.

Let $x = u_i c u_i$ and $x' = u_j c u_j$ be two different elements of W_m ($i \neq j$). We pick y as u_i^r and z as u_j^r . Then, $yxz = u_i^r u_i c u_i u_j^r \in \mathbf{PromisePAL}_{\text{yes}}$ and $yx'z = u_i^r u_j c u_j u_j^r \in \mathbf{PromisePAL}_{\text{no}}$. Thus, no bounded-error 2PFA can solve **PromisePAL** due to Fact 1. \square

Freivalds and Karpinski [18] presented a slightly different fact to prove that **PAL** cannot be recognized by any bounded-error sublogarithmic probabilistic Turing machine (PTM).

B The proof of Theorem 3

Instead of calling the 2QCFA (for **PAL**) of Ambainis and Watrous [3] as a whole, we call a subroutine of this algorithm and use it in our new algorithm. We refer to it as $\mathcal{AW}_{\mathcal{PAL}}$ and its details are as follows. $\mathcal{AW}_{\mathcal{PAL}}$ uses a quantum register with 3 states, i.e. $|q_1\rangle, |q_2\rangle, |q_3\rangle$, which is set to $|q_1\rangle$ at the beginning. $\mathcal{AW}_{\mathcal{PAL}}$ reads the given input, say $w \in \{a, b\}^*$, from left to right twice. In the first pass, it applies U_σ to the quantum register for each $\sigma \in \{a, b\}$, and, in the second pass, it applies U_σ^{-1} to the quantum register for each $\sigma \in \{a, b\}$, where

$$U_a = \frac{1}{5} \begin{pmatrix} 4 & 3 & 0 \\ -3 & 4 & 0 \\ 0 & 0 & 5 \end{pmatrix} \text{ and } U_b = \frac{1}{5} \begin{pmatrix} 4 & 0 & 3 \\ 0 & 5 & 0 \\ -3 & 0 & 4 \end{pmatrix}.$$

The quantum state of $\mathcal{AW}_{\mathcal{PAL}}$ after the two passes is given by:

$$|\psi_f\rangle = \mathbf{U}_{|w|}^{-1} \mathbf{U}_{|w|-1}^{-1} \cdots \mathbf{U}_{w_2}^{-1} \mathbf{U}_{w_1}^{-1} \mathbf{U}_{|w|} \mathbf{U}_{|w|-1} \cdots \mathbf{U}_{w_2} \mathbf{U}_{w_1} (1 \ 0 \ 0)^T$$

The two passes are followed by a measurement in the computational basis. From [3], we know that

- If $w \in \text{PAL}$, then $|\psi_f\rangle = |q_1\rangle$, and so, the measurement result is the first state.
- If $w \notin \text{PAL}$, then $|\psi_f\rangle \neq |q_1\rangle$ and the second or third state is observed with a probability at least $25^{-|w|}$.

Now, we describe an exact rational restarting rtQCFA for $\mathcal{EXACT}_{\mathcal{TWINPAL}}$, a modified version of **PromisePAL**. i.e.

PromiseTWINPAL = (**PromiseTWINPAL_{yes}**, **PromiseTWINPAL_{no}**), where

- **PromiseTWINPAL_{yes}** = $\{ucucvcv|u, v \in \{a, b\}^+, |u| = |v|, u \in \text{PAL}, \text{ and } v \notin \text{PAL}\}$, and
- **PromiseTWINPAL_{no}** = $\{ucucvcv|u, v \in \{a, b\}^+, |u| = |v|, u \notin \text{PAL}, \text{ and } v \in \text{PAL}\}$.

Let $w = ucucvcv$, a member of **PromiseTWINPAL**, be the input such that $u, v \in \{a, b\}^*$ and $|u| = |v|$. $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ reads the input only from left to right in an infinite loop. A single iteration of this loop proceeds as follows. $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ starts with quantum state $(1 \ 0 \ 0)^T$. Then, $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ splits the computation into two branches, $branch_1$ and $branch_2$. For this purpose, $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ applies \mathbf{U}_a to the quantum register and then measures it in the computational basis. $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ continues with $branch_1$ (resp., $branch_2$) with probability $\frac{16}{25}$ (resp., $\frac{9}{25}$) when the first (resp., the second) state is observed. In $branch_1$, $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ executes $\mathcal{AW}_{\mathcal{PAL}}$ for v on vcv . Similarly, in $branch_2$, $\mathcal{AW}_{\mathcal{PAL}}$ is executed for u on ucu . Here $\mathcal{AW}_{\mathcal{PAL}}$ reads v (or u) twice by scanning vcv (or ucu). $\mathcal{EXACT}_{\mathcal{TWINPAL}}$ gives its decision based on the outcome of $\mathcal{AW}_{\mathcal{PAL}}$ at the end. If the second or third state is observed in $branch_1$ (resp., $branch_2$), then w is accepted (resp., rejected) by $\mathcal{EXACT}_{\mathcal{TWINPAL}}$. Otherwise, the computation continues with the next iteration. The analysis of a single iteration is as follows:

- Suppose that $w \in \text{PromiseTWINPAL}_{\text{yes}}$. Then, $branch_1$ ends in a state different than $|q_1\rangle$ with probability at least $25^{-|v|}$ since $v \notin \text{PAL}$, and, $branch_2$ ends in state $|q_1\rangle$ always since $u \in \text{PAL}$. Therefore, a decision is given only in the first branch. That is, w is accepted with a probability at least $16 \cdot 25^{-|v|-1}$.
- Suppose that $w \in \text{PromiseTWINPAL}_{\text{no}}$. Then, $branch_1$ ends in state $|q_1\rangle$ always since $u \in \text{PAL}$, and, $branch_2$ ends in a state different than $|q_1\rangle$ with probability at least $25^{-|u|}$ since $u \notin \text{PAL}$. Therefore, a decision is given only in the second branch. That is, w is rejected with probability at least $9 \cdot 25^{-|u|-1}$.

Therefore, the members of **PromiseTWINPAL_{yes}** (resp., **PromiseTWINPAL_{no}**) are accepted (resp., rejected) exactly in expected exponential time as stated in Theorem 3.

C The proof of Theorem 4

In order to prove that 2PFAs (or sublogarithmic space PTMs) cannot solve promise problem **PromisE**TWIPAL with bounded-error, we use a reduction technique used in [40]. It was shown that $\text{TWIN} = \{ucu \mid u \in \{a, b\}^*\}$ cannot be recognized by bounded-error 2PFAs. The proof is based on the following contradiction: Assume **TWIN** can be recognized by a bounded-error 2PFA. Then **PAL** can also be recognized by a bounded-error 2PFA. Since **PAL** is known to be not recognizable by bounded-error 2PFA [15], we end up with a contradiction and our assumption must have been incorrect.

Suppose that a bounded-error 2PFA, say \mathcal{P} , can solve **PromisE**TWIPAL. Then we can construct a bounded-error 2PFA, say \mathcal{P}' , that solves **PromisE**PAL as follows. For a given input $w = ucv$, a member of **PromisE**PAL, \mathcal{P}' can simulate \mathcal{P} on $w' = ucucvcv$ and gives the same output as \mathcal{P} . The simulation is straightforward: After leaving a c or an end-marker, the head of \mathcal{P} is on the first or the last symbol of u or v . \mathcal{P}' can easily keep a track of such leaves by its internal states, and so, it can place its head at the corresponding place of u or v , respectively. So, \mathcal{P}' can solve **PromisE**PAL with bounded error. But, as shown previously, this is impossible. We can derive the same result also for bounded-error sub-logarithmic space PTMs.

D Exact 2PFAs

From an exact 2PFA, we can obtain two two-way nondeterministic finite automata, say \mathcal{N}_1 and \mathcal{N}_2 , that give the decision of “acceptance” for yes instances and no instances, respectively. So, there are two rtDFAs \mathcal{D}_1 and \mathcal{D}_2 with the same behaviour. Then, we can obtain the desired rtDFA by taking the tensor product of \mathcal{D}_1 and \mathcal{D}_2 . For each member of the promise problem, either \mathcal{D}_1 or \mathcal{D}_2 gives the decision of “acceptance”.

E The proof of Theorem 5

We start by analyzing the long term behaviour of a single branch of $\mathcal{E}\mathcal{X}\mathcal{A}\mathcal{C}\mathcal{T}_{\text{TWINPAL}}$ (see Appendix B). Let $w = ucucvcv$ be a member of **PromisE**TWIPAL_{yes} such that $u, v \in \{a, b\}^*$ and $|u| = |v| = n$. Suppose that we execute branch_1 $T = 25^n$ times on w . The overall accepting probability is given by

$$\sum_{i=1}^T 25^{-n} (1 - 25^{-n})^{i-1} = 25^{-n} \left(\frac{1 - (1 - 25^{-n})^T}{1 - (1 - 25^{-n})} \right) = 1 - \left(1 - \frac{1}{25^n} \right)^{25^n}.$$

Since $\lim_{x \rightarrow \infty} \left(1 - \frac{1}{x} \right)^x = \frac{1}{e}$ and $\left(1 - \frac{1}{x} \right)^x$ is less than $\frac{1}{e}$ for every $x \in \mathbb{Z}^+$, the overall accepting probability is always greater than

$$1 - \frac{1}{e}.$$

The same result can also be obtained for the overall rejecting probability of $branch_2$ when executed on a member of $\text{PromisePAL}_{\text{no}}$, say $w = ucucvcv$, 25^n times, where $u, v \in \{a, b\}^*$ and $|u| = |v| = n$.

Remember the definition of our promise problem:

$$\text{EXPPromiseTWINPAL} = (\text{EXPPromiseTWINPAL}_{\text{yes}}, \text{EXPPromiseTWINPAL}_{\text{no}}),$$

where

- $\text{EXPPromiseTWINPAL}_{\text{yes}} = \{(ucucvcv)^t | u, v \in \{a, b\}^+, |u| = |v|, u \in \text{PAL}, v \notin \text{PAL}, \text{ and } t \geq 25^{|u|}\}$, and
- $\text{EXPPromiseTWINPAL}_{\text{no}} = \{(ucucvcv)^t | u, v \in \{a, b\}^+, |u| = |v|, u \notin \text{PAL}, v \in \text{PAL}, \text{ and } t \geq 25^{|u|}\}$.

We now give a Las Vegas rational rtQCFA $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ for the problem. Let $w = (ucucvcv)^t \in \text{EXPPromiseTWINPAL}$, where $u, v \in \{a, b\}^*$, $|u| = |v| > 0$, and $t \geq 25^{|u|}$. At the beginning, the computation is split into two branches with probabilities $\frac{16}{25}$ and $\frac{9}{25}$ (as described before). In the first (resp., the second) branch, $branch_1$ (resp., $branch_2$) of $\mathcal{EXACTWIPAL}$ is implemented on each block of $ucucvcv$ until the end of input. $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ gives the decision of “don’t know” if the input has not been accepted or rejected. Using our analysis earlier in the section, we can conclude that the probabilities of decisions given by $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ are given by:

- Any member of $\text{EXPPromiseTWINPAL}_{\text{yes}}$ is accepted with probability at least $\frac{16}{25} (1 - \frac{1}{e})$. So, $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ gives the decision “don’t know” with a probability no more than $\frac{9}{25} + \frac{16}{25e}$.
- Any member of $\text{EXPPromiseTWINPAL}_{\text{no}}$ is rejected with probability at least $\frac{9}{25} (1 - \frac{1}{e})$. So, $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ gives the decision “don’t know” with a probability no more than $\frac{16}{25} + \frac{9}{25e}$.

Executing many copies of $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ in parallel, we can increase both the accepting and rejecting probabilities arbitrary close to 1. The probability of “don’t know” then gets closer to 0. To eliminate “don’t answer”, we can restart $\mathcal{LV}_{\mathcal{EXPTWIPAL}}$ whenever we obtain a “don’t know” answer. Thus, we can obtain a linear expected time rational exact restarting rtQCFA solving EXPPromiseTWINPAL .

F The proof of Theorem 6

We show that no bounded-error rtPFAs can solve EXPPromiseTWINPAL . Suppose that there exists such a rtPFA called \mathcal{P} that works with error probability $\frac{1}{3}$. Then, by using \mathcal{P} , we can construct a bounded-error 2PFA, say \mathcal{P}' , for PromiseTWINPAL . Let $w = ucucvcv$ be the input such that $u, v \in \{a, b\}^+$ and $|u| = |v| = n > 0$. \mathcal{P}' executes \mathcal{P} on $(ucucvcv)^i$ for each $i > 0$ and then follows the decision of \mathcal{P} with a certain probability. More specifically, \mathcal{P}' can easily execute \mathcal{P} on $(ucucvcv)^i$ for each $i > 0$ in an infinite loop. That is, after the i^{th} iteration,

\mathcal{P} is fed $(ucucvcv)^i$. At this point, before continuing with the next iteration, the decision of \mathcal{P} on $(ucucvcv)^i$ is implemented with probability 25^{-16n} . This is performed by reading $O(n)$ symbols. With the remaining probability, $1 - 25^{-16n}$, \mathcal{P}' continue with the next iteration, i.e. it feeds another $ucucvcvc$ to \mathcal{P} . Note that the last state of \mathcal{P} at the end of the i^{th} iteration is recorded by the internal states of \mathcal{P}' and \mathcal{P} is set to this state again just before the next iteration starts.

Let $T = 25^n$. Assume that $w \in \text{PromiseTWINPAL}_{\text{yes}}$. Then, after the $(25^n)^{th}$ iteration, we know that \mathcal{P} gives the decision of acceptance with probability at least $\frac{2}{3}$. In the worst case, we can assume that \mathcal{P} gives the rejection decision before the $(25^n)^{th}$ iteration. Then the overall accepting probability of \mathcal{P}' can be calculated as follows.

The input is rejected with probability

$$\sum_{i=1}^{T-1} 25^{-16n} (1 - 25^{-16n})^{i-1} < 1 - \sqrt[16]{\frac{1}{e}}$$

before the T^{th} iteration. So, the computation continues from the T^{th} iteration with probability at least $\sqrt[16]{\frac{1}{e}}$, which is greater than 0.9. After this point, the input is accepted with a probability at least $\frac{2}{3}$ and so the overall accepting probability is always greater than 0.6. Similar analysis holds for the members of $\text{PromiseTWINPAL}_{\text{no}}$, and so, the rejecting probability of such members also exceeds 0.6. Thus, \mathcal{P}' can solve EXPPromiseTWINPAL with bounded error. This is a contradiction.

G The proof of Theorem 7

2QCFA's can recognize EQ in polynomial expected time [3] with one-sided bounded-error. The given algorithm executes two phases, a quantum and then a classical one, in an infinite loop. We will use only the first phase and call it $\mathcal{AW}_{\mathcal{EQ}}$ which operates on a single qubit $(|q_1\rangle, |q_2\rangle)$ and uses only algebraic transitions. The input is read by $\mathcal{AW}_{\mathcal{EQ}}$ in realtime mode, and if it is $a^m b^m$, then $|q_1\rangle$ is observed exactly, and if it is $a^m b^n$ ($m \neq n$), then $|q_2\rangle$ is observed with a probability at least $\frac{1}{2(m-n)^2}$. Thus, similar to the previous algorithm, we can easily obtain a quadratic expected time exact algebraic restarting rtQCFA algorithm solving PromiseEQ .

Recently Yakaryılmaz [39] presented a linear-time one-sided bounded-error one-way QFA (1QFA) algorithm for the language $\{a^n b a^n\}$ where the model uses *quantum head*. This algorithm can be easily modified to use only rational numbers (see [35,36]). Then, we can follow that PromiseEQ can be solved by a Las Vegas rational 1QFA in linear time, and so by an exact rational two-way QFA's in linear expected time (see [41]).

H The proof of Theorem 8

In [16,14], Dwork and Stockmeyer showed that 2PFAs or $o(\log \log n)$ -space PTMs can recognize a non-regular language only in super-polynomial expected time. Their proofs follows from some technical facts. We will review the related facts and then modify them for our aim.

For a given language $L \subseteq \Sigma^*$ and a length n , two strings whose lengths are no more than n , say w and w' , are called n -dissimilar, i.e. $w \approx_{L,n} w'$, if there exists a string $u \in \Sigma^*$ satisfying $|wu| \leq n$ and $|w'u|$ such that

$$wu \in L \text{ if and only if } w'u \in \bar{L}.$$

We call u as a separator of w and w' . $N_L(n)$ is the maximum number of pairwise n -dissimilar strings.

Fact 2 (Lemma 4.3 on Page 1017 in [14]) *For every $\epsilon < \frac{1}{2}$, there are positive constants α_ϵ and η_ϵ such that, if a 2PFA \mathcal{M} having c states recognizes the language L with error bound ϵ and within expected time $T(n)$, then*

$$(\alpha_\epsilon c (\log T(n) + \log cn))^{n^2} \geq N_L(n) \text{ for all } n \geq \eta_\epsilon. \quad (2)$$

The proof was given based on a contradiction such that if Equation 2 does not hold then two n -dissimilar strings, say w and w' , cannot be distinguished by \mathcal{M} . That is, if wu is accepted by \mathcal{M} with a probability at least $1 - \epsilon$, then $w'u$ is accepted by \mathcal{M} with a probability at least $\frac{1}{2}$, where u is a separator of w and w' . The good news is that the proof still works if we focus on promise problems after modifying the dissimilarity relation as follows.

For a given promise problem $P = (P_{\text{yes}}, P_{\text{no}})$ defined on Σ and a length n , two strings whose lengths are no more than n , say w and w' , are called n -dissimilar, i.e. $w \approx_{P,n} w'$, if there exists a string $u \in \Sigma^*$ satisfying $|wu| \leq n$ and $|w'u|$ such that

$$wu \in P_{\text{yes}} \text{ if and only if } w'u \in \overline{P_{\text{no}}}.$$

$N_P(n)$ is the maximum number of pairwise n -dissimilar strings. Now, we can rephrase the fact above for promise problems.

Theorem 11. *For every $\epsilon < \frac{1}{2}$, there are positive constants α_ϵ and η_ϵ such that, if a 2PFA \mathcal{M} having c states solves the promise problem P with error bound ϵ and within expected time $T(n)$, then*

$$(\alpha_\epsilon c (\log T(n) + \log cn))^{n^2} \geq N_P(n) \text{ for all } n \geq \eta_\epsilon. \quad (3)$$

It is clear that if $N_L(n)$ (or $N_P(n)$) is not less than n^δ for some $\delta > 0$ for infinitely many n , then the corresponding bounded-error 2PFA cannot have a sub-exponential expected time. As described in [14], if we augment a 2PFA with a $s(n)$ -space work tape, a $s(n)$ -space PTM, then we can safely replace c with $c(n) = 2^{ds(n)}$ in Equations 2 and 3, where d is a constant depending on

the machine. Then, after simple calculations, we can follow our sub-exponential expected time statement also for bounded-error $o(\log \log n)$ -space PTMs.

Now, we show that $N_{\text{PromiseEQ}}(n) \geq \Omega(n)$ which is sufficient to conclude that **PromiseEQ** cannot be solved by any bounded-error $o(\log \log n)$ -space PTMs in polynomial expected time. We define a set of m strings $\{a^i \mid 1 \leq i \leq m\}$. These m strings are $(3m+1)$ -dissimilar since for any two different $w = a^i$ and $w' = a^j$, there exists $u = ba^i ba^j$ such that

$$wu = a^i ba^i ba^j \in \text{PromiseEQ}_{\text{yes}} \text{ and } w'u = a^j ba^i ba^j \in \text{PromiseEQ}_{\text{no}}.$$

I The proof of Theorem 9

The behaviour of a rtPFAs over a single letter can be modelled as a Markov chain [2,29,26]. We utilize a fact given in [29].

Fact 3 (Theorem 6 on Page 486 of [15]) *For any $m \times m$ stochastic matrix A , there is an integer d such that the limit*

$$\lim_{i \rightarrow \infty} (A^d)^i$$

exists. Furthermore, there are $k \geq 1$ positive integers d_1, \dots, d_k , which depend on A such that $d_1 + \dots + d_k \leq m$ and $d = \text{lcm}(d_1, \dots, d_k)$.

Let \mathcal{P} be an m -state rtPFA defined over a unary alphabet $\Sigma = \{a\}$ and A be its stochastic transition matrix defined for a . Based on the above fact, we can conclude that the series defined by the accepting probabilities of the strings

$$\{a^d, a^{2d}, a^{3d}, \dots\}$$

has a limit, say a_1 . Similarly, that of

$$\{a, a^{d+1}, a^{2d+1}, a^{3d+1}, \dots\}$$

has also a limit, say a_2 . In a similar fashion, we can define (at most) d limiting accepting probabilities: $\{a_1, \dots, a_d\}$. Let $\delta > 0$ be sufficiently small such that $(a_i - \delta, a_i + \delta)$ and $(a_j - \delta, a_j + \delta)$ do not overlap unless $a_i \neq a_j$, where $1 \leq i < j \leq d$. We name each interval $(a_i - \delta, a_i + \delta)$ as a region called r_i , where $1 \leq i \leq d$. Let M_δ be a sufficiently big integer such that the accepting probability of each string longer than M_δ lies in one of the regions defined above and a^{M_δ} lies in a_1 . Thus, the accepting probabilities of strings

$$a^{M_\delta}, a^{M_\delta+1}, a^{M_\delta+2}, \dots, a^{M_\delta+d-1}, a^{M_\delta+d}, a^{M_\delta+d+1}, \dots$$

lie in regions

$$r_1, r_2, r_3, \dots, r_d, r_1, r_2, \dots,$$

respectively. We now observe that the regions are visited in a cycle. Any unary deterministic finite automaton can enter a similar cycle for long strings as well.

Ambainis and Yakaryılmaz showed that [4] any rtDFA can solve EVENODD^k only if $\gcd(d, 2^k) \neq \gcd(d, 2^{k+1})$. That is, the length of the cycle must be a multiple of 2^{k+1} . The same argument is also true for our case. The details are given below.

Suppose that \mathcal{P} can solve EVENODD^k with an error bounded $\epsilon \in (0, \frac{1}{2})$ and $m < 2^{k+1}$. Without loss of generality, we can also assume that if a_i is in $(\epsilon, 1 - \epsilon)$, then r_i does not contain both ϵ and $1 - \epsilon$, where $1 \leq i \leq d$. Therefore, it is clear that the members of $\text{EVENODD}_{\text{yes}}^k$ and $\text{EVENODD}_{\text{no}}^k$ longer than N_δ must lie in the different regions. Let $L = 2^l = \gcd(d, 2^{k+1})$ for some non-negative integer l . Since $m < 2^{k+1}$, l can be at most k . (Remember that $d = \text{lcm}(d_1, \dots, d_k)$ and each d_i is less than m , where $1 \leq i \leq k$ for some $k \geq 1$. Therefore, each d_i can contain at most l 2(s) as its factor(s).) Therefore, $\gcd(d, 2^k)$ is equal to 2^l as well. The accepting probabilities of strings

$$\{a^{i2^k} \mid 2^k \geq N_\epsilon \text{ and } i > 0\}$$

lie in $t = \frac{d}{2^l}$ different r -regions, i.e. $R = \{r'_1, \dots, r'_t\}$. More precisely,

- the accepting probability of $a^{1 \cdot 2^k}$ lies in r'_1 ,
- the accepting probability of $a^{2 \cdot 2^k}$ lies in r'_2 ,
- the accepting probability of $a^{3 \cdot 2^k}$ lies in r'_3 ,
- ...
- the accepting probability of $a^{t \cdot 2^k}$ lies in r'_t ,
- the accepting probability of $a^{(t+1) \cdot 2^k}$ lies in r'_1 ,
-

We can divide these regions into two classes, namely *yes regions* and *no regions*: yes regions are closer to $1 - \epsilon$ and no regions are closer to ϵ . Here a yes region corresponds to a member of $\text{EVENODD}_{\text{yes}}^k$ and a no region corresponds to a member of $\text{EVENODD}_{\text{no}}^k$. Therefore, each class must not be empty. Now, we focus on only the members of $\text{EVENODD}_{\text{no}}^k$: The accepting probabilities of strings

$$\{a^{i2^k} \mid 2^k \geq N_\epsilon \text{ and } i \text{ is an odd positive integer}\}$$

lie in also $t = \frac{d}{2^l}$ different r -regions that are actually a subset of R . More precisely,

- the accepting probability of $a^{1 \cdot 2^k}$ lies in r'_1 ,
- the accepting probability of $a^{3 \cdot 2^k}$ lies in r'_3 ,
- the accepting probability of $a^{5 \cdot 2^k}$ lies in r'_5 ,
- ...
- the accepting probability of $a^{t \cdot 2^k}$ lies in r'_t ,
- the accepting probability of $a^{(t+2) \cdot 2^k}$ lies in r'_2 ,
- the accepting probability of $a^{(t+4) \cdot 2^k}$ lies in r'_4 ,
- ...
- the accepting probability of $a^{(t+t-1) \cdot 2^k}$ lies in r'_{t-1} ,
- the accepting probability of $a^{(t+t+1) \cdot 2^k}$ lies in r'_1 ,
-

Thus, yes regions must be empty. This is a contradiction. This is why $\gcd(d, 2^k)$ must be different than $\gcd(d, 2^{k+1})$. It can be only possible when d is a multiple of 2^{k+1} . As pointed in [4], it is straightforward that EVENODD^k can be solvable by a 2^{k+1} -state realtime deterministic finite automaton.